12

Application of CAPS Modeling to Strategy Competition and Flexibility in Discourse Comprehension

Susan R. Goldman, Sashank Varma, and Julio Ortega
Learning Technology Center
Vanderbilt University

Final Report
August 15, 1992

This project was supported by Office of Naval Research Cognitive Science Program Grant N00014-91-J-1769. Approved for public release, distribution unlimited. Reproduction in whole or in part is permitted fro any use of the United States Government.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 8/15/92	3. REPORT TYPE AND DATES COVERED Final Report 5/1/91 - 4/30/92

4. TITLE AND SUBTITLE
Application of CAPS Modeling to Strategy Competition and Flexibility in Discourse Comprehension

5. FUNDING NUMBERS
Grant N00014-91-J-1769

6. AUTHOR(S)
Susan R. Goldman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Learning Technology Center
Vanderbilt University
Box 45, Peabody
Nashville, TN 37203

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Cognitive Science Program (1142CS)
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unrestricted

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Behavioral data obtained by Goldman and Saul (1990) suggest the need to incorporate competition among reading strategies into text comprehension models. REREADER is a computational model that does that. The model is implemented within the Collaborative Activation-Based Production System (CAPS) architecture (Just & Carpenter, 1990). In CAPS, productions must accumulate sufficient activation to fire and their activations fluctuate depending on other aspects of the system. Total activation is limited by working memory capacity; when this limit is reached active elements lose activation so additional processing can occur. In REREADER, reading strategies are represented as productions and competition among them is governed by activation thresholds. REREADER operates on a linked set of input propositions that represent the text. Processing occurs in 4 phases. The first three create a semantic network of propositions linked on the basis of argument overlap and propositional embedding and having differential levels of activation. In phase 4, if sufficient levels of coherence obtain (determined by an evaluation function that includes a motivation parameter), REREADER continues to process new input; otherwise, a rereading strategy operates. Initial tests of the system indicate the plausibility of the computational model. Additional development will incorporate prior knowledge and more sophisticated strategy competition mechanisms.

14. SUBJECT TERMS
Computational modeling
Text Comprehension

15. NUMBER OF PAGES
34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT

Abstract

Behavioral data obtained by Goldman and Saul (1990) suggest the need to incorporate competition among reading strategies into text comprehension models. REREADER is a computational model that does that. The model is implemented within the Collaborative Activation-Based Production System (CAPS) architecture (Just & Carpenter, 1992). In CAPS, productions must accumulate sufficient activation to fire and their activations fluctuate depending on other aspects of the system. Total activation is limited by working memory capacity; when this limit is reached active elements lose activation so additional processing can occur. In REREADER, reading strategies are represented as productions and competition among them is governed by activation thresholds. REREADER operates on a linked set of input propositions that represent the text. Processing occurs in 4 phases. The first three create a semantic network of propositions linked on the basis of argument overlap and propositional embedding and having differential levels of activation. In phase 4, if sufficient levels of coherence obtain (determined by an evaluation function that includes a motivation parameter), REREADER continues to process new input; otherwise, a rereading strategy operates. Initial tests of the system indicate the plausibility of the computational model. Additional development will incorporate prior knowledge and more sophisticated strategy competition mechanisms.

PART I. INTRODUCTION AND BACKGROUND

Goldman and Saul (1990a) proposed a Strategy Competition Model for text processing to account for the individual flexibility in reading that they observed using specially developed Macintosh software (Goldman & Saul, 1990b). This work was supported by the Cognitive Science Program of the Office of Naval Research specifically to examine similarities and differences in the comprehension and reasoning processes of linguistic minority and native English speakers (Goldman, Durán, Murray, Saul, & Smith, 1989).

The initial formulation of the Strategy Competition Model outlined several components, including knowledge of rhetorical devices for signalling topic change and importance, processes to achieve local coherence, and monitoring and evaluation processes. Readers' verbal reports indicated that reading strategies were related to various kinds of comprehension problems (Goldman, 1991). Students reported comprehension problems at multiple levels - word, sentence, across sentences - and repair strategies of several types. Several conclusions were drawn from that research:

- Individuals have a repertoire of repair strategies.
- The likelihood that a particular strategy is selected depends on
  - the type of comprehension problem
  - the relative strengths of the repair strategies
- Strength values are related to
  - past history with different kinds of comprehension problems and resolution of them
  - task factors, including motivation
  - text factors (especially surface characteristics)

Testing the implications of the strategy competition model requires the ability to conduct computational modeling. With such a technique it is possible to manipulate factors that are extremely difficult to manipulate in human subjects. These include prior knowledge, including factual and strategic; past history with texts and structural feature knowledge derived from these experiences; and motivation. By constructing a comprehender whose "cognitive" features are "known," specific implications of the model can be tested. The results of the computational model can then be compared to behavioral data. When good "fits" are obtained, it is often plausible to assume a mapping between the computational model characteristics and human learners.

Goals and Objectives

At the inception of this project, two architectures, Kintsch's Construction-Integration model (CI) (Kintsch, 1988, in press) and Just and Carpenter's Collaborative Activation Production System (CAPS) (Just & Carpenter, 1992; MacDonald, Just, & Carpenter, 1992) were being used to model language processing of single sentences or of short, 2 or 3 sentence passages. Each seemed like an appropriate starting points for a computational model that might be adapted to processing longer, connected discourses (text passages) such as those found in textbooks, magazines, or newspapers. The CI model uses a propositional representation for text and specifies a matrix of connections among them. The matrix is integrated in the connectionist manner and the resulting strengths among the propositions can be used to predict strength in working term and in long term memory (Kintsch, 1988, in press; Kintsch & Welsh, in press; Kintsch, Welsh, Schmalhofer & Zimny, 1990).

The CAPS architecture is a production-based system in which the conditions accumulate activation until their "firing" threshold is reached. Total activation in working memory is limited by the working memory "cap." The cap varies from individual to individual. As text is processed, working memory resources are allocated; once the allocation reaches the cap, activation values of the elements in working memory decay proportionately to their activation levels. The CAPS model as used by Just and Carpenter is most sensitive to identifying points of high processing load during comprehension.

The two types of architectures are differentially applicable to the behavioral data obtained by Goldman and Saul (1990a; Goldman, 1991, 1992). The CI model can be used to predict behavioral recall data but it is not obvious how it would predict reading and rereading strategies of the sort observed by Goldman and Saul (1990a). The CAPS architecture appeared to be extendable to processing text (as opposed to single sentences) and to have the potential to predict recall and rereading strategies. However, the extension was a non-trivial project. That extension was the goal of the current project. It has been realized in REREADER, described in Part II of this final report. A second goal of this project was to develop a user interface so that sophisticated knowledge of programming was not necessary to change the parameters of the system. The User Interface is described in Part III of this final report.

PART II
REREADER: A COMPUTATIONAL MODEL OF STRATEGIC TEXT COMPREHENSION

The CAPS Architecture

One of the appealing aspects of the CAPS architecture was that it appeared to have the potential to address both the working memory issues and strategy competition. The CAPS architecture developed by Just and Carpenter (1990) permits one to build "comprehenders" that have (i) different knowledge but the same "working memory resource" capacities, or (ii) different capacities but the same knowledge. We could then observe the effects on comprehension processes and on the products of that process and determine how various aspects of the system contribute.

A key aspect of the CAPS architecture is that preconditions on actions can exist at *levels of activation* rather than as merely present or absent. So a "reader model" may behave differently depending on the other elements in working memory, the activation limit for working memory, and the available activation. In the CAPS architecture when the activation limit is reached, information already in working memory has to share the activation with new elements that enter working memory; the "older" elements lose activation to the newer ones. When elements fall below a minimum level, they are no longer functional in working memory - they can't connect up with new information. They are, for all intents and purposes, gone.

Just and Carpenter have been using a parser written in the CAPS architecture - CCREADER- to simulate high versus low span processing of sentences differing in assumed processing load. For example, they examined a set of relative clause sentences that have "momentary" structural ambiguity if the relative pronoun is deleted (McDonald, Just, & Carpenter, 1992). (Span is reflected in the value specified as the upper limit on activation.) They collected time per word data for the words in the two sentences The lawyer addressed by the judge was held in contempt and The lawyer that addressed the judge was held in contempt. In the first case the reader does not know whether *addressed* is the main verb of the sentence or a verb participle in the reduced relative clause. CCREADER parses and assigns grammatical and case role relations among the sentence elements. By manipulating the activation limits, Just and Carpenter provide evidence that in the absence of high demand situations, there is little difference in the way these sentences are read by high and low span readers. However, at points of high processing load in the sentence, high and low span readers show different empirical patterns; and "systems" that have high compared to lower working memory activation limits simulate the effects quite successfully (Just & Carpenter, 1992).

REREADER

When we began to adapt the CAPS architecture to processing text, we initially tried to expand the CCREADER parser to convert surface structure input into propositions. We abandoned this effort and decided instead to focus on establishing connections among propositionalized sentences and on the repair strategies used when initial linking procedures failed. The user can manipulate the characteristics of the "reader" through a user-friendly interface. REREADER is the prototype reader model that we have developed. Our prototyping efforts were modest and restricted to one informational passage entitled Distance and shown in Table 1.

Characteristics of the System

1. Representation of the input text.

The system operates on a text base like the one used by Kintsch in implementing the CI model. A sample is given in Table 2 using the first sentence of the Distance passage. Sentences are parsed into a verbal predicate with optional and obligatory arguments (predicate 1) along with the relevant concepts (existence nodes), and modification propositions. A fourth type of proposition relates predicates, e.g., cause, contrast, etc. In REREADER the propositions are connected using three linking strategies: argument overlap, propositional embedding, and relational reference. There are several variables that can be adjusted: (a) The weights, or initial activation values assigned to the propositions; (b) The linking strategies the system operates with; and (c) The weights assigned to the links. The four types of propositions that REREADER uses - verbal predicates, concept propositions, modification propositions, and relational predicates - are assigned weights. Table 2 shows one set of weights that we have investigated: verbal predicates are assigned a weight of 2 and everything else is assigned weights of 1. Links between propositions are weighted equally (a value of 1). The activation values for propositions and links may be modified through the user interface.

2. System Knowledge.

There are two kinds of "spaces" in the system: one corresponds to human working memory and the other to long term memory. Working memory processes incoming input. The results of each processing cycle are accumulated in a "long term memory" representation. The system also has in long term memory a dictionary. It contains information classifying the predicates for the specific text, a set of equivalence elements (translators), and a set of procedural elements such as the propositional linking strategies and reading strategies.

The variables related to system knowledge are the following: the Activation Cap, i.e, the capacity of working memory; the amount of prior knowledge in the dictionary; procedural knowledge available to the system; and motivation, which affects how the production system behaves.

The working memory cap proportionately degrades all elements when the system is operating at the limit. The cap for each run is set through the interface (we have experimented with caps in the 30 to 125 unit range thus far). Different caps will affect the life of an input proposition in working memory because degradation to working memory elements occurs when the cap is reached. For example, when reading the second sentence of a passage, the working memory resources might be sufficient to handle creation of all requested elements at the targeted initial activations without stealing activation from elements preexisting in working memory. On the fourth sentence, however, all available activation might be allocated and further requests, perhaps to build within-sentence links, would result in the degradation of all elements (i.e. their activations would be scaled back). Over many processing cycles at the cap, a working memory element would lose most of its activation and fall below threshold, thus becoming invisible (unusable) in further processing unless it is re-activated.

3. Reading Strategies

The system "knows" four reading strategies: Read forward, read backwards, skim backwards to a problem, skim forward to sentence that initiated a reread. (Skim - orthographic representations of words get some low levels of activation but no propositional linking is done and no propositions are reentered in working memory.) These strategies are based on the behavioral data of Goldman and Saul (1990a).

The strategies specified in the Strategy Competition Model involved different numbers of sentences - sometimes a reader read just one sentence back, sometimes three or four sentences back to a specific sentence. In REREADER, these different kinds of strategies are a result of evaluations of the current strategy after the processing of each sentence. These evaluations depend on several aspects of coherence and the motivation level of the system. The evaluation function is described below.

4. How REREADER Works.

REREADER operates in four conceptual phases illustrated in Figure 1.

Phase I. Reads the first sentence as a string of propositions and assigns activation levels according to a weighting framework specified by the user. For the example set shown in Table 2, weights of 1 are assigned to concepts, modification, and relational propositions and 2 to main predicates of the sentence. If the working memory "cap" or limit is exceeded, elements in working memory "lose" activation proportionally until the new elements can realize the appropriate values to be recognized by the system.

The next two phases deal with link construction.

Phase II. Creates within — sentence links. For the sentence shown in Table 2, P1 would be connected to P4 and to P5. This part of the system operates just like the CI model of Kintsch. In **Phase III**, between sentence links are created according to argument overlap, propositional embedding and relational reference rules.

Phase IV. This phase proceeds in two steps. First, REREADER computes measures of text coherence using the following five pieces of information:

1. Number of links to the first sentence of the passage if the current sentence is the first sentence of a paragraph. A higher value equals greater coherence.

2. Number of links to the first sentence of the paragraph if the current sent-

ence is not the first sentence of a paragraph. A higher value equals greater coherence.

3. Number of links to previous sentence (superseded by 2 or 1). So if the current sentence is second in the paragraph, procedure 2 will operate and 3 will not. For the third sentence in a paragraph, procedure 3 will operate as will procedure 2 but they will be counting different links.

4. Number of propositional arguments referenced in the current sentence that are still present in working memory.

5. Number of propositional arguments referenced in the current sentence that are missing from working memory. This measure reflects problems integrating the current input with elements already in working memory; in other words, failure to find overlap.

The second step is to evaluate the current reading strategy, using the measures of text coherence as inputs, and returning the "next" reading strategy. "Read forward" will be selected if there is a positive value returned by the evaluation function. "Mark problem" will be selected if there is a zero value returned by the evaluation function. "Read backward" will be selected if there is a negative value returned by the evaluation function. How REREADER proceeds given each of these outcomes is described following a discussion of the evaluation function.

The formula for the Evaluation Function reflects competition between reasons for believing adequate coherence exists and reasons for believing it does not, with motivation weighting the latter. More precisely, it is the sum of measure 1 (above) plus measure 2 plus (measure 3 divided by 5) less (measure 5 + motivation). The reason measure 3 is divided by 5 is so that local links will not totally dominate reading. The denominator of 5 allows the system to take into account more global aspects of coherence as well as coherence with the just — prior sentence.

Motivation has a direct effect on the likelihood of rereading. With high motivation going back will be more likely than if motivation is low. In fact, with this evaluation function, two readers with the same number of missing arguments would behave differently depending on their motivation levels. (Note that measure 4, the number of propositional arguments active in working memory, does not figure in the evaluation function. We spent two weeks searching for a suitable function, one which seemed plausible and produced reasonable behavior. Altnough the present evaluation function does not include measure 4, replacing measure 5 in the evaluation function given above with (measure5 ÷ (measure4 + measure5)), i.e., the proportion of missing arguments, yields similar performance.)

Strategy selection depends on the evaluation function in the following manner. If the evaluation function returns a positive or zero value strategy selection is relatively trivial. A positive value means that there is sufficient connection between the current sentence and propositions still active in working memory. In other words, coherence is okay so reading continues forward. A zero value indicates that the sentence is partially connected but that there are a number of potential connections that were not made. The sentence will be marked as a problem. If sentences are marked, the marker becomes important if the system decides to read backwards. In the future, we intend to use these kinds of markers for decisions at the end of text about general level of understanding and comprehension.

A negative value results in the system reading backwards. This is essentially a judgment that the current sentence lacks coherence with the propositions active in working memory. If the system decides to read backwards, there are two possibilities: it can skim back to a marked-problem sentence or it can read backward sequentially from the current sentence.

a. The Marked-problem case: This case occurs when there is a proposition active in working memory that has been marked as a problem. (These will tend to be within 3 or 4 sentences back for most normal activation caps.) The system skims back to that sentence, reads it and then evaluates. If the evaluation function returns a positive value or a zero value the problem was resolved sufficiently. REREADER skims forward to the sentence that initiated the rereading and continues reading forward (i.e., reads in the next sentence). However, if the evaluation function returns a negative value, the problem remains; the system then proceeds backwards from the marked problem sentence. It stops going backwards when the evaluate function returns read forward. (Note that in this case, REREADER skips over sentences between the marked problem sentence and the sentence that initiated the rereading. This is reasonable because in most cases if those "skipped over" sentences had had a lot of coherence with the sentence that initiated the rereading there would not have been a coherence problem to begin with, i.e., the "skipped over" propositions were available for integration the first time around.)

If there are multiple, marked sentences active in working memory, REREADER goes through them sequentially in the following way: If there are multiple marked problems in memory, go to nearest one and read it. Evaluate. If there is still lack of coherence and there are more marked problems go to the next one. Evaluate. If coherence is still lacking and there are no more marked problems, read backwards.

b. Read backward sequentially. This case occurs when there are no or no more "marked-problem" sentences in working memory. After each sentence is read, there is an evaluation. When Evaluate does not return read backwards, skim forward again to sentence that initiated the backtrack and read the next sentence.

In cases of read backwards, if the first sentence of the passage is reached and the evaluation function is still returning read backwards, REREADER skims back to the sentence that initiated the rereading, marks it as a problem, and resumes reading forward. One additional special case may occur: During rereading it is possible for the REREADER to "lose it's starting place" (initiating sentence). If it does, it begins reading forward from the current sentence.

In summary, the evaluation function and rereading work in the following way: REREADER reads forward and at the end of each sentence evaluates how well it integrates with information active in working memory. If there are small problems, REREADER marks the sentence and continues reading forward. If there are larger problems, REREADER proceeds backwards. If any sentences are marked, REREADER skims back and reads each one starting at the closest one. If none are marked, REREADER reads backwards sequentially. When the evaluation function no longer returns read backwards, the problem has been solved; and REREADER skims forward to where the problem occurred and resumes reading forward.

Long term memory strengths are summed on the basis of the activation levels for each proposition at the conclusion of each input cycle. These are cumulated and saved for later reference.

Initial Runs of the Prototype

Initial runs of REREADER have produced some interesting variations of reading traces. The Distance passage was parsed into 167 propositions following the principles that Kintsch uses for constructing a propositional text base. This parse was the input to the REREADER simulation.

Figure 2 provides two reading traces generated by our REREADER system. Comparison of the traces illlustrates the effects of working memory capacity when motivation is held constant and relatively high. Trace75 is a low capacity simulation and Trace 125 a high capacity (75 and 125 units of activation, respectively). The traces show the path through the Distance passage or the order in which the sentences were processed. Cycle by cycle information can be used in conjunction with the path data to gain a detailed understanding of the effects of capacity differences on processing activity. These sources of difficulty can subsequently be compared to sources of difficulty experienced by human readers.

The first thing to notice is that the higher capacity trace has fewer backtracks than the lower capacity; as well the sentences that trigger a backtrack and the nature of the backtracking differ. For example, the low capacity trace shows that reading proceeds forward until sentence 11. At this point, there is insufficient coherence, largely because a main referent in the sentence,*distance* , has fallen below threshold. There is a backtrack to sentence 7 which had been marked as problematic previously. However, the missing concept proposition is not found in sentence 7 and there is systematic rereading back to the beginning of the passage. When sentence 1 is reread, the missing concept proposition is reactivated allowing REREADER to skim ahead to sentence 12 and continue reading forward. When sentence 12 is read it is marked as problematic and in sentence 13 there is another missing referent that causes a negative evaluation function to be returned. REREADER reads backward to sentence 3, where the rereading activates the missing concept. Having resolved the problem, REREADER resumes reading at the sentence that triggered the backtrack.

However, on the very next sentence, 14, the concept *distance* is once again needed but because of low capacity, the intervening processing has resulted in *distance* falling below threshold once again. REREADER reads back to sentence 1 again to reactivate the missing argument; it then picks up at sentence 15. At sentence 16, the concept *movement* is missing and REREADER starts to read backwards. When sentence 14 is reread, links are created between its propositions and those in sentence 16. These links produce sufficient coherence for the evaluation function to return a positive value, and no further rereading goes on, despite the fact that the missing *movement* concept was not reactivated. This illustrates the nondeterministic, competitive nature of the system. Sentence 17 is processed and the below-threshold activation of the concept*distance* again produces rereading behavior: REREADER skims back to the first sentence marked as problematic (15) but the problem is not resolved and rereading continues back to sentence 1 where *distance* is reactivated. However, by the time REREADER has gotten back to sentence 1, many of the original propositions from sentence 17 have fallen below threshold. Thus, sentence 17 never really gets successfully integrated with the rest of the text base and is marked as problematic. Having gotten to the beginning of the text, the system skims back down to sentence 18 and resumes reading forward. A number of propositions in sentence 18 first occurred in sentence 17. However, because of the long reread triggered by sentence 17, many of these are below threshold when sentence 18 is read. A reread of 17 reactivates the missing arguments and satisfies the coherence constraints permitting reading to proceed with sentence 19.

In summary, with low capacity the theme concept of the passage, *distance,* fell below threshold by sentence 11. This created several rereading events. Coherence could be established through rereading so long as the number of intervening sentences was kept relatively low. When almost the entire passage was reread, capacity was exceeded and when the missing concept was located, the sentence that had initiated the rereading had lost too much activation for a link to be established. The lengthy reread also interfered with coherence between contiguous sentences and the system had to reread 17 after it read 18 in order to establish coherence.

In contrast to the persistent need to reread (5 occasions) in the low capacity simulation, the higher capacity simulation produces only 2 rereads and these tend to be more efficient because problematic markers are retained longer in the higher capacity simulation. Trace125 in figure 2 shows that no rereading occurs until sentence 13, where the high capacity simulation encountered the same problem as the low capacity, a missing referent. REREADER skims back to sentence 7 that had been marked as problematic and, not finding the missing referent, reads back until it is activated (sentence 3). The system continues reading sentence 14 but the concept *distance* is below threshold and it rereads until *distance* is found in sentence 1. Reading continues at sentence 15 and, in contrast to the low capacity trace, no other problems were encountered.

Although these traces produced plausible reading and rereading behavior and sensible differences between high and low capacity "readers" there were a couple of features of the traces that were somewhat at variance with the behavioral data collected by Goldman and Saul (1990a, 199c). In both the Trace75 and Trace125 simulations the majority of the rereads lasted ten to fifteen sentences. However, the behavioral data indicated that the about 50% of rereads were one or two sentences back. By lowering the activation cap and motrivation level, we hoped to shift REREADER's behavior into a similar pattern. A lower capacity would increase the decay rate of propositions, causing propositions to lose arguments which lay in sentences just a few sentences back. Because this would result in a tremendous amount of rereading we adjusted the motivation level as well. We wanted to create simulation conditions that would lose more propositions than in our first runs but would be more tolerant of small comprehension problems: Only when coherence was really poor would REREADER reread and it would resume reading forward when minimal rather than absolute coherence was achieved. We had observed that in Trace75 and Trace125 the simulation would reread when even a single argument was missing and did not resume reading forward until all missing arguments had been re-activated.

We conducted new simulations using 30 (low) and 50 (high) for capacity values and setting motivation levels of 0 and 1. Trace1 in Figure 3 shows the reading path for a low capacity, high motivation simulation. The overall shape of the path indicates more localized backtracking than in Trace75 or Trace125. In addition, when rereading occurred, the simulation often satisfied coherence requirements within one or two sentences back. This rereading left some "missing" referent concepts among the propositions but satisfied enough of the possible links that the evaluation function returned a positive value and forward reading resumed. One of the interesting things about Trace1 is that the simulation did not keep returning to the *distance* concept from sentence 1 the way it did in Trace75 and Trace125 to some degree. Rather, in a number of sentences that had *distance* as an argument, sufficient coherence was achieved that no links were created between the concept *distance* (from sentence 1) and the arguments. This would lead to a representation of the information that is less tightly connected to a passage theme, an outcome that seems plausible for readers with relatively low working memory capacities.

The results for the high capacity simulation, Trace2 shown in Figure 4, differ from Trace1 in sensible and plausible ways. Trace2 REREADER goes twice as far as Trace1 before backtracking (at sentence 14) where there is insufficient coherence with the active propositions. However, three prior sentences have been marked as problems. The simulation goes back to a marked sentence and then reads all the way back until sufficient overlap is established with the concepts in sentence 5. Then reading proceeds forward from the point of the backtrack. There is only one other rereading incident - at the end of the passage, although each of the three prior sentences were marked as problems. After reading sentence 19, REREADER goes back and rereads sentence 18, where it finds a missing argument. Having satisfied coherence requirements, the simulation skims 19 again and quits. Thus, in contrast to Tracel, Trace2 simulation indicates 2 rereads. When compared to Trace125, which had a higher motivation value, Trace2 is similar except that it quit reading earlier (after cycle 40 as compared to 65 for Trace125).

To summarize, the prototype CAPS REREADER Model allowed us to implement all but one of the five components of the Strategy Competition Model. We have simulated procedural rules for establishing local coherence, procedural rules for establishing global coherence, strategy selection based on competition among the strengths of propositions and procedures in working memory, and coherence monitoring and evaluation. The REREADER prototype does not use procedural rules for reacting to features of the text, however we can see how to implement this feature in a "smarter" REREADER. In that version of the system, which is currently under development, prior knowledge will also be incorporated and we hope to explore some mechanisms that simulate how individuals add new information to a permanent knowledge base.

Part III

User Interface for REREADER

The User Interface for REREADER guides the user through the operation of REREADER. Running REREADER involves specifying a text file and "dictionary" for the text, setting parameter values (including a path), and selecting desired output files. Each of these is briefly described here. Expanded information on the user interface is available in the CAPS REREADER Manual.

Specifying the text file and "dictionary"

The text to be read must be in text file format but can be produced by any number of word processing applications. A specific format is required as illustrated here for a one sentence text:

```
(convert-text '(
(sentence 1)
(start paragraph 1)
distance is simply the space between 2 points
(1 is (pp4,pp5)
(2 is^between (pp5 pp6))
(3 quant (pp6 2))
(4 distance)
(5 space)
(6 points)

))
```

The format is basically similar to that used frequently in propositional analysis (e.g., Kintsch, 1988; Turner & Green, 1978). The first element in the parentheses is the proposition number. This is followed by the verbal for the proposition. We combine certain surface elements into one by linking them, as in *is^between*.

Sentence elements appearing by themselves, as in line 8, indicate concepts. The sentence number and sentence (see lines 2 and 4 in the example) are included in the file. Line 3 indicates that this sentence starts a paragraph. Inclusion of this information makes it possible to subsequently implement different activation depending on the position of a sentence in a paragraph.

The "Dictionary"

This is essentially REREADER's knowledge base. In the present implementation, it contains information about the assignment of each proposition to one of four types (predicate, relational, concept/existence, or modification). Equivalences among concepts are also specified here. Future versions will also incorporate propositions representing lexical and schematic knowledge relevant to the passage.

Setting Parameter Values

CAPS REREADER depends on a number of parameter values, some of which may be modified through the User Interface and other of which may not be.

Parameters Accessible through the User Interface

The parameters that *may be* altered by the user are shown in Figure 5, which is a copy of the CAPS REREADER Initialization screen. The user may decide to force REREADER to proceed through a text using a specific sentence order or permit it to run normally. In the former case, a path file must be specified in advance and referenced on the initialization screen. Other parameter values include the activation cap, motivation, constituent activations, strategy link options and coherence criteria options. The default values for constituent activations were described above. Strategy link options allow the user to choose the set of linking rules that operate on a given run. The default is for all three to operate (argument overlap, propositional embedding and relational reference). At the same time, the default link activation value of 1 can be altered. The default for coherence criteria is for all five measures described above to be computed and used in the evaluation function. However, the user can deselect one or more of these indices of coherence.

Parameters Modifiable through Changes to the Source Code

Several system variables are not accessible from the User Interface and can be modified only by directly changing the source code. This code is entirely contained in the text files INITIALIZE.L (Psylisp code) and REREADER.L (CAPS productions and commands). All parameters described in this section can be found in the latter file.

1. Activation-Capping Scheme: CAPS REREADER supports three activation-capping schemes. Each was developed after flaws in a predecessor were discovered. REREADER, like most CAPS production systems, runs under the third one. This is specified early in the source file with the command (turn pre-spew-adjustment 3.0).

2. Activation Threshold: Working memory elements in CAPS have activations. For an element to match a production, it must match symbolically and have an activation greater than the specified threshold. When not explicitly stated in a production, this threshold defaults to 0.1. This can be changed by modifying the line (turn default-production-threshold 0.1) early in the source file.

3. Tracing: REREADER's path through a passage is reported at the sentence level. That is, every time a sentence is processed, several bits of information are displayed including the sentence being scanned, whether it's being read or skimmed, whether it was reached by moving forwards or backwards, the values of the measures of text coherence and evaluation function, what course of action REREADER decides to take next based on the value of the evaluation function, the markers of problematic sentences that are still active (above the Activation Threshold described above), and the total number of CAPS cycles used to process the

sentence and passage so far. The last two quantities hint at the underlying truth that REREADER takes tens of internal cycles to process each sentence. The processing over these cycles in the form of production firing and the activation constrainment can be viewed in detail by changing selected 'off' parameters in the following command (found early in the source file) to 'on': (turn trace-productions off trace-cycles off trace-all-spews off trace-cap off).

4. Starting Information: REREADER currently begins processing by reading forward from the first sentence of the passage. The initial task, reading or skimming, can be changed by altering the RHS action (<add> (current-task read)) in the first production, called *initialize-system*. The initial direction is set by the RHS action (<add> (current-direction forward)) in the same production. To begin processing at a sentence other than the first one, modify the RHS action (<add> (current-phase (start (<tok> sentence 1)))) in the next production, called *read-freely*. This modification has no effect when a fixed path is supplied because the first sentence on the path is the first sentence that will be read.

5. Phase Sequencing: CAPS is fully parallel, capable of firing all matched productions on each cycle. This allows several lines of processing to occur at the same time. REREADER currently processes sentences by cycling through several phases. The sequence in which these phases occur (or co-occur) is specified by working memory elements of the form (:ltm *task direction phase-i* :becomes *phase-j*) where the italics indicate variables. These elements are created in the production *read-freely* and *read-path* depending on whether REREADER is evaluating the coherence of the text after each sentence and deciding what to do next based on the evaluation or reading a predetermined path through the passage. For instance, freely reading a sentence while moving forward requires cycling through six phases, one at a time, as indicated by the following working memory elements:

```
(:ltm read forward start :becomes scan-sentence)
(:ltm read forward scan-sentence :becomes link-with-sentence)
(:ltm read forward link-within-sentence :becomes link-between-sentences)
(:ltm read forward link-between-sentences :becomes fire-demons)
(:ltm read forward fire-demons :becomes evaluate-current-reading-strategy)
(:ltm read forward evaluate-current-reading-strategy :becomes end).
```

To change the "read forward" behavior so that scan-sentence, link-within-sentence, and link-between-sentences occur simultaneously, change the second and third elements above to:

```
(:ltm read forward start :becomes link-within-sentence)
(:ltm read forward start :becomes link-between-sentences).
```

To flip the order in which within-sentence and between-sentence links are created, alter the second, third, and fourth elements above to:

```
(:ltm read forward scan-sentence :becomes link-between-sentences)
(:ltm read forward link-between-sentences :becomes link-within-sentence)
(:ltm read forward link-within-sentence :becomes fire-demons)
```

There are three additional parameters that, like the first five, are not currently modifiable from the User Interface. However, we plan to provide access to them from the User Interface in a future version.

6. Linking Boost: When a link is created, a small amount of activation is kicked back to the propositions being joined. This number is set by the RHS action (<init-propositional-link-boost> 0.1) in the production *initialize-system*.

7. Number of Words to Scan: REREADER scans a sentence in one pass if it is skimming and two passes if it is reading. The first pass (which both modes share)

activates an orthographic representation of each word (called a percept). The additional pass associated with reading is used to activate the propositions representing the semantic content of the sentence. Initially, it took one CAPS cycle to activate a percept and one to activate a proposition. Because propositions presumably require deeper processing, and thus more time to activate, REREADER was changed to activate multiple percepts on each cycle. The RHS action (<number-of-words-to-scan> 3) in the production *initialize-system* sets this number. The default setting ensures that percepts are activated three times as quickly as propositions.

8. Evaluation Function Threshold: The evaluation function returns a number based on the measures of text coherence. REREADER compares this number with the Evaluation Function Threshold and returns an evaluation (read on, mark problem, proceed backwards) depending on whether it is less than, equal to, or greater than the threshold. To set this threshold, alter the (<init-new-reading-strategy-threshold> 2) RHS action in the production *read-freely*.

The default values for these variables were determined based on procedures for tuning the operation of REREADER, following standard simulation techniques.

Selecting Output Files

The Initialization screen requests information regarding the names of files in which trace information for the simulation run will be stored. These files are used to construct a variety of reports, as indicated in Figure 6, Report Selection Menu. As can be seen on this menu, a variety of analyses can be requested, including a listing for each cycle as well as various summaries. As well, an automatic edit feature based on activation strength (threshold) is included. Only propositions that exceed the threshold are included in the report information. As Figure 6 indicates data are formattee in the Excel spreadsheet and can be directly imported to that application.

References

Goldman, S. R., Durán, R. P., Murray, J., Saul, E. U., & Smith, M. (1989, August). *Reasoning and comprehension processes of linguistic minority persons learning from text.* Final Report to Cognitive Science Program, Office of Naval Research. Santa Barbara, CA: University of California.

Goldman, S. R. & Saul, E. U. (1990a). Flexibility in text processing: A strategy competition model. *Learning and Individual Differences, 2,* 181-219.

Goldman, S. R. & Saul, E. U. (1990b). Applications for tracking reading behavior on the Macintosh. *Behavior Research Methods, Instruments, and Computers, 22,* 526-532.

Goldman, S. R., & Saul, E. U. (1990c, November). *Paragraphing and task effects on reading strategies.* Paper presented at the Psychonomics Society Meetings, New Orleans, LA.

Just, M.A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review, 99,* 122-149.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review, 95,* 163-182.

Kintsch, W. (in press). How readers construct situation models for stories: The role of syntactic cues and causal inferences. In A.F. Healy, S. Kosslyn, & R. M. Shiffrin (Eds.), *Essays in honor of William K. Estes* (pp.). Hillsdale, NJ: Erlbaum.

Kintsch, W.,Welsch, D., Schmalhofer, F., & Zimny S. (1990). Sentence memory: A theoretical analysis. *Journal of Memory and language, 29,* 133-159.

Kintsch, W., & We[illegible]ch, D. (in press). The construction-integration model: A framework for studying memory for text. In W. E. Hockley & S. Lewandowsky

(Eds.), *Relating theory and data: Essays on human memory.* Hillsdale, NJ: Erlbaum.

MacDonald, M. C., Just, M. A., & Carpenter, P. A. (1992). Working memory constraints on the processing of syntactic ambiguity. *Cognitive Psychology, 24,* 56-98.

Turner, A. & Greene, E. (1978). The consturction and use of a propositional text base. *JSAS Catalog of Selected Documents in Psychology, 8,* 58.

Table 1

Distance Passage

Distance is simply the space between two points. Measured with a standard unit such as the kilometer or mile, the result is called absolute distance and is what most persons probably think of when they think of distance. However, there is also something called relative distance, which is when absolute distance is influenced by other factors.

There are a number of factors that affect distance measurement. Economic distance is measured in relation to the cost of movement from one place to another. There is a cost involved in any movement, in terms of either money or energy. The cost of shipping something by water is usually about one-tenth the cost over land, despite the fact that land routes are usually shorter.

Distance can be measured relative to time. Maps that use travel time instead of mile markers as the units of measure distort the usual spatial relations among locations. Travel time from a single point, such as the central business district, to a location 10 miles north may be the same as travel time to a location 30 miles south.

Psychological distance is measured relative to our perceptions, which may vary. What may seem like a long trip to some individuals may seem short to others. Even the same route going and coming can seem different to a single traveler, depending on road conditions, time of day, or anticipation of the end of the trip.

Distance can be measured relative to direction of movement. When 2 points are located at different elevations, movement from point A to point B may not be as easy as from B to A. The mileage between the points may be equal but uphill movement is harder than downhill.

Distances in geographic space can seem longer due to friction with obstacles. For example, large crowds or heavy traffic are considered friction because they slow down our movement. Often we are willing to travel farther in order to reduce friction, such as when we go to a suburban mall rather than to a downtown store to avoid traffic and crowds.

Table 2

Sample Propositional Parse for the First Sentence of the Distance Passage

Sentence: Distance is simply the space between two points.

Proposition	Type of Proposition	Activation
p0 simply(p1)	modification	1
p1 Is(p4,p5)	verbal predicate	2
p2 is^between(p5,p6)	modification	1
p3 quant(p6,2)	modification	1
p4 distance	concept	1
p5 space	concept	1
p6 points	concept	1

Figure 1: Sentence Processing Procedure

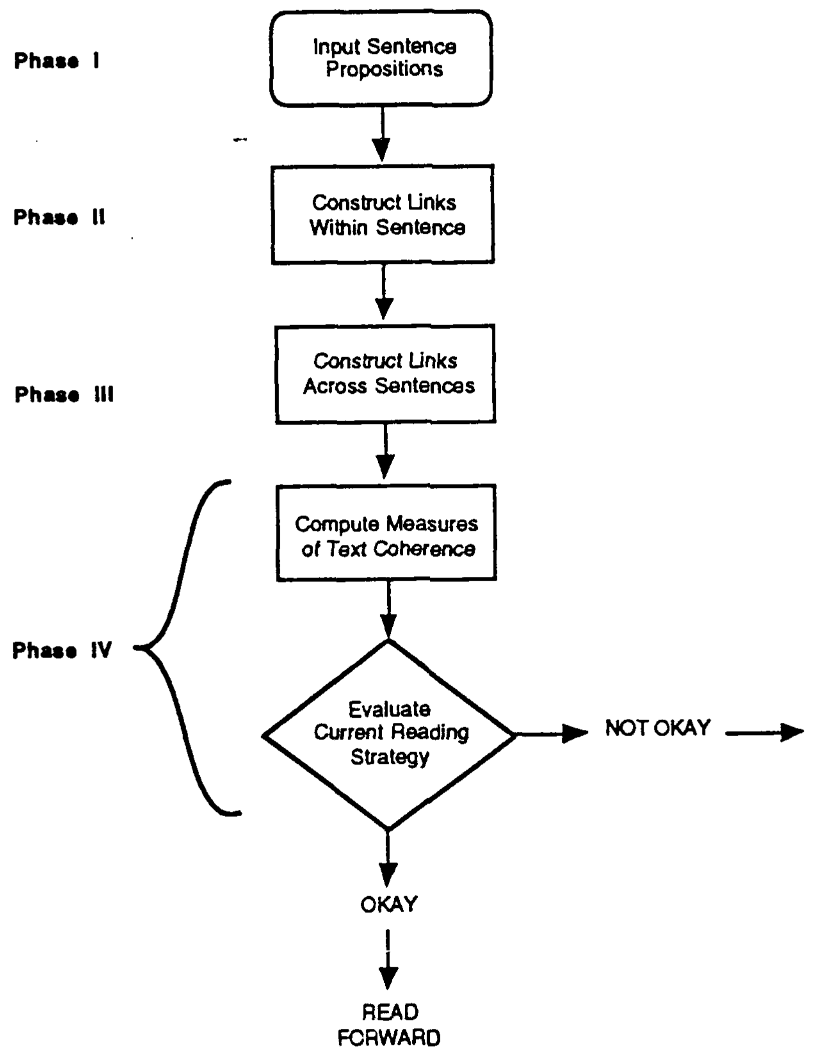
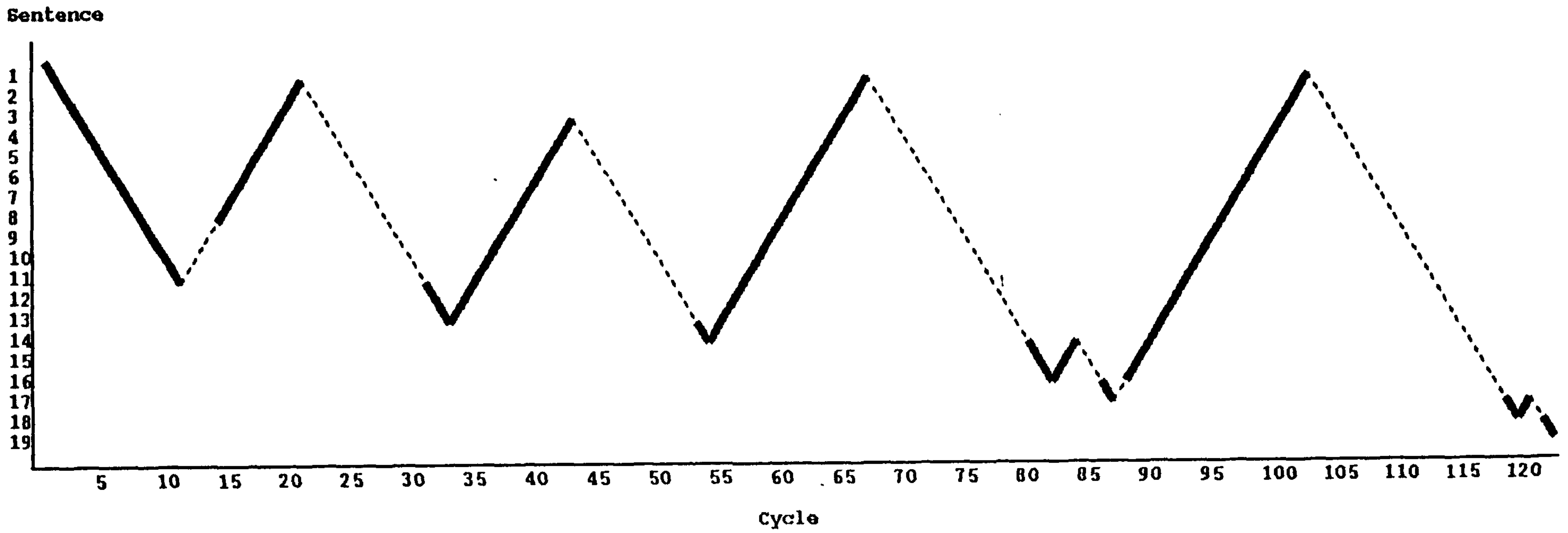
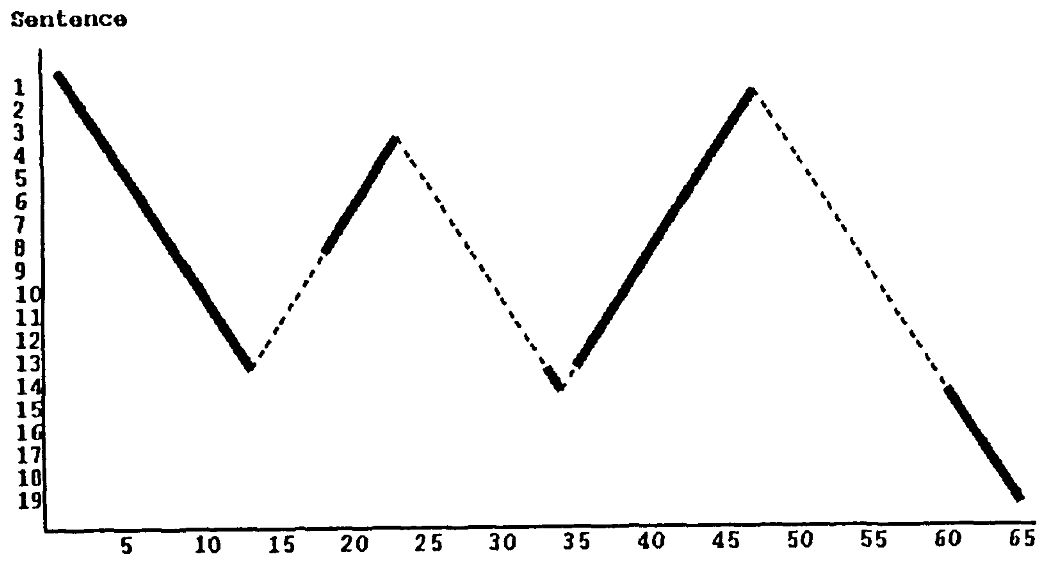

Phase I
Input Sentence Propositions
Phase II
Construct Links Within Sentence
Phase III
Construct Links Across Sentences
Phase IV
Compute Measures of Text Coherence
Evaluate Current Reading Strategy
NOT OKAY
OKAY
READ FORWARD

Figure 2



Trace 75



Trace125

Figure 3: Trace1

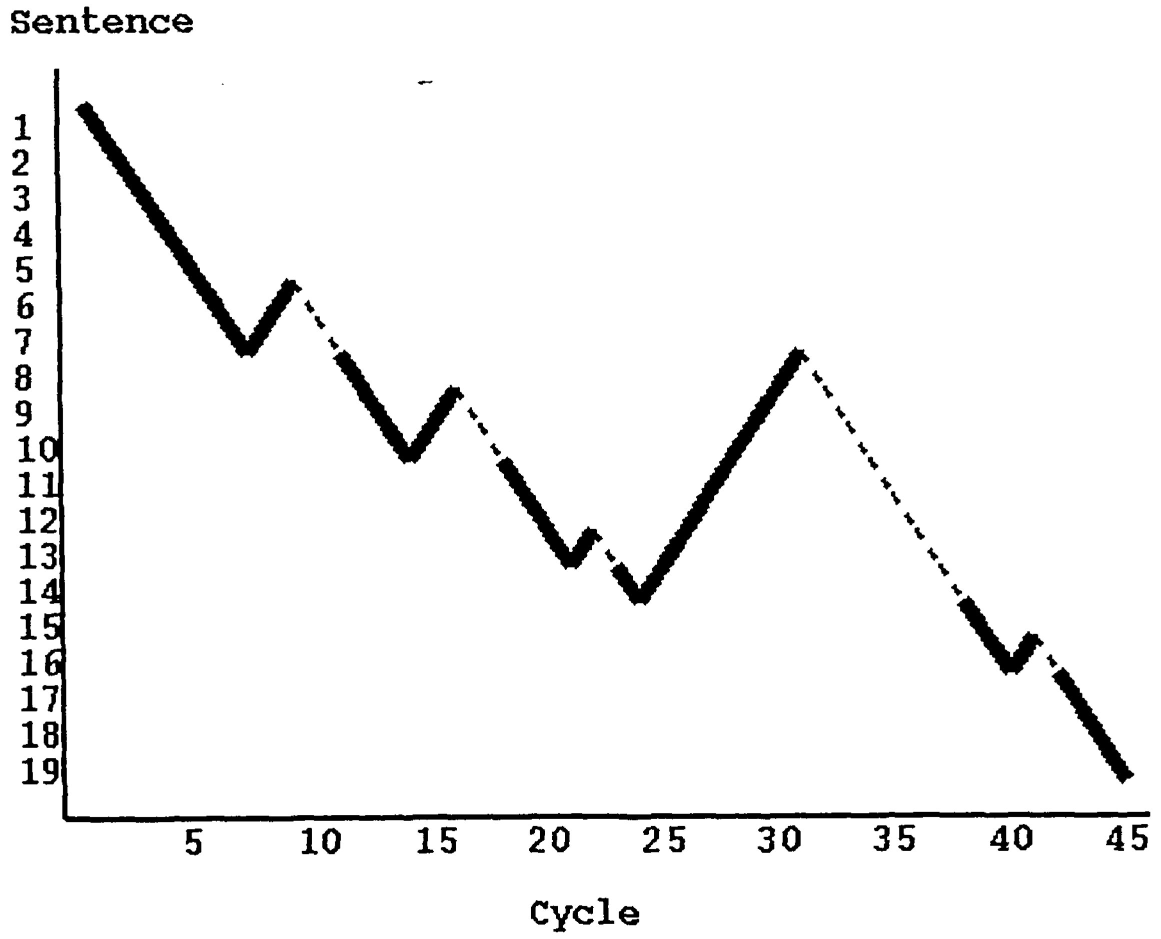
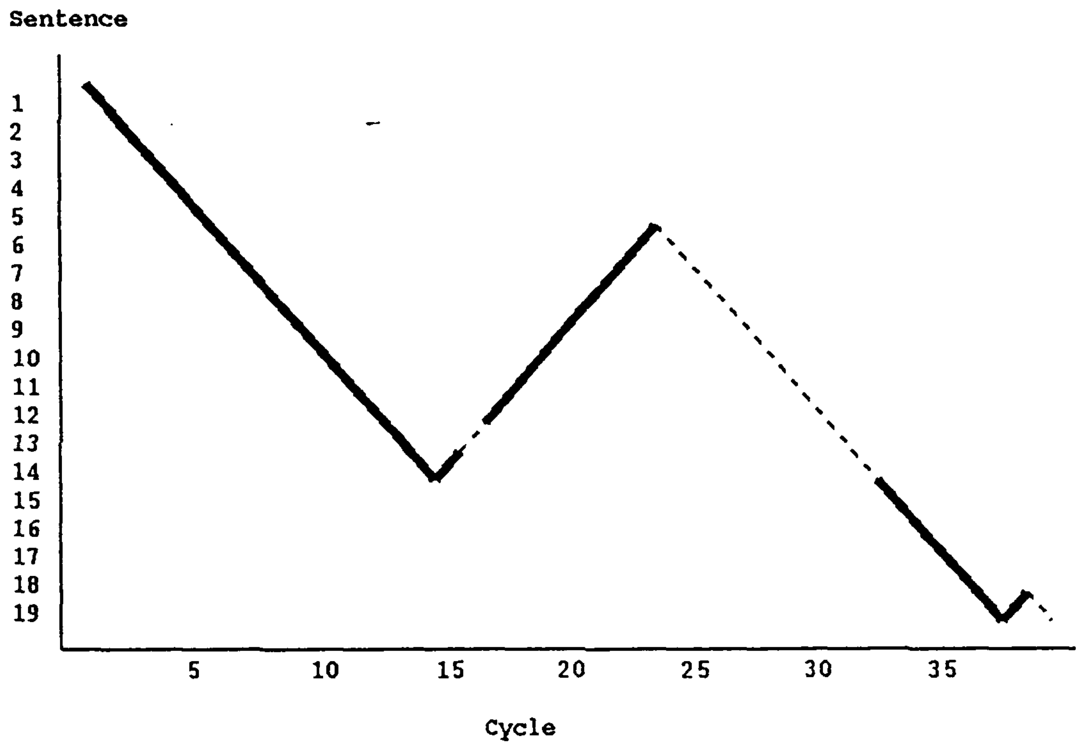
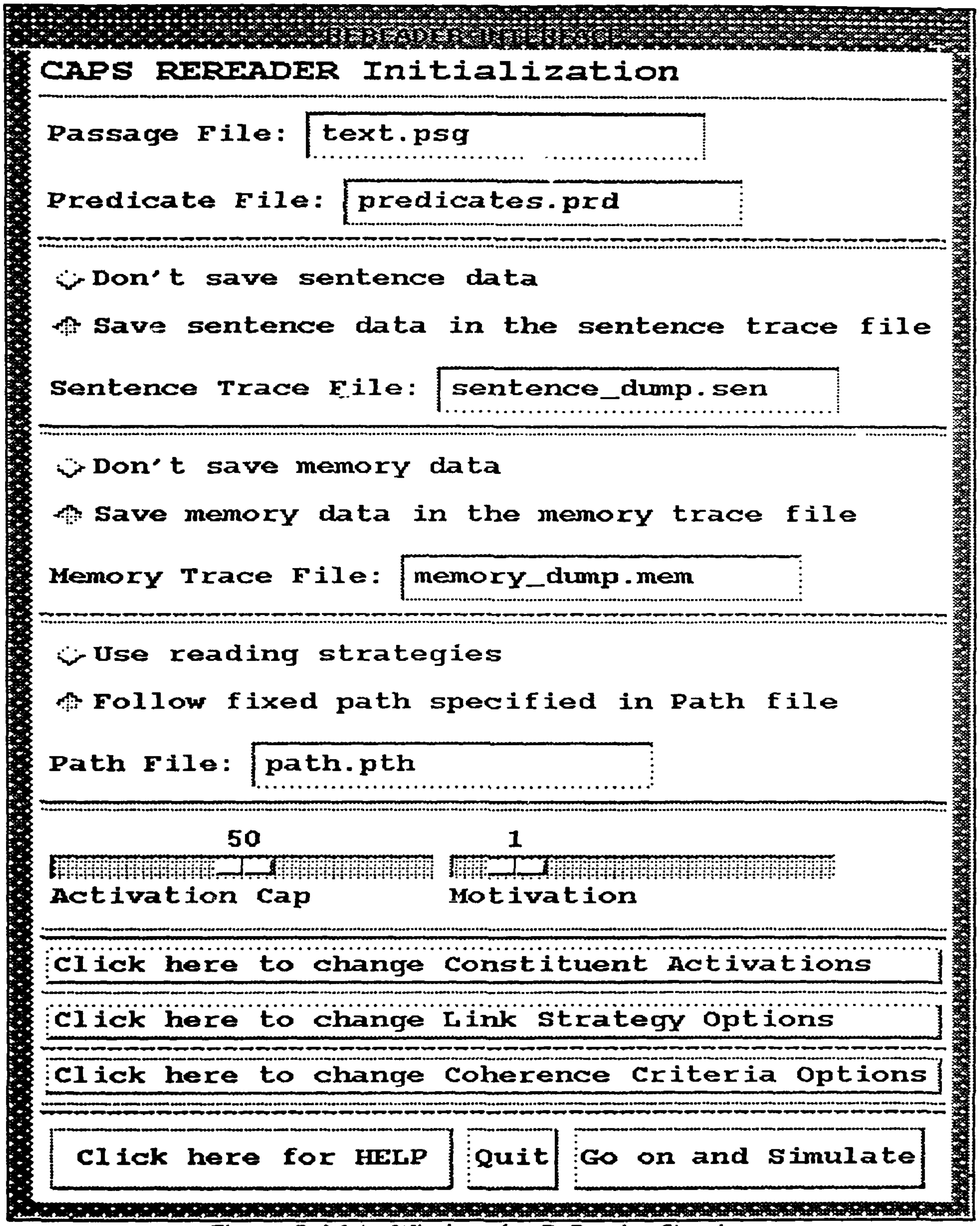


Figure 4: Trace2





REREADER INTERFACE

CAPS REREADER Initialization

Passage File: text.psg

Predicate File: predicates.prd

Don't save sentence data

[✓ Save sentence data in the sentence trace file]

Sentence Trace File: sentence_dump.sen

Don't save memory data

[✓ Save memory data in the memory trace file]

Memory Trace File: memory_dump.mem

Use reading strategies

[✓ Follow fixed path specified in Path file]

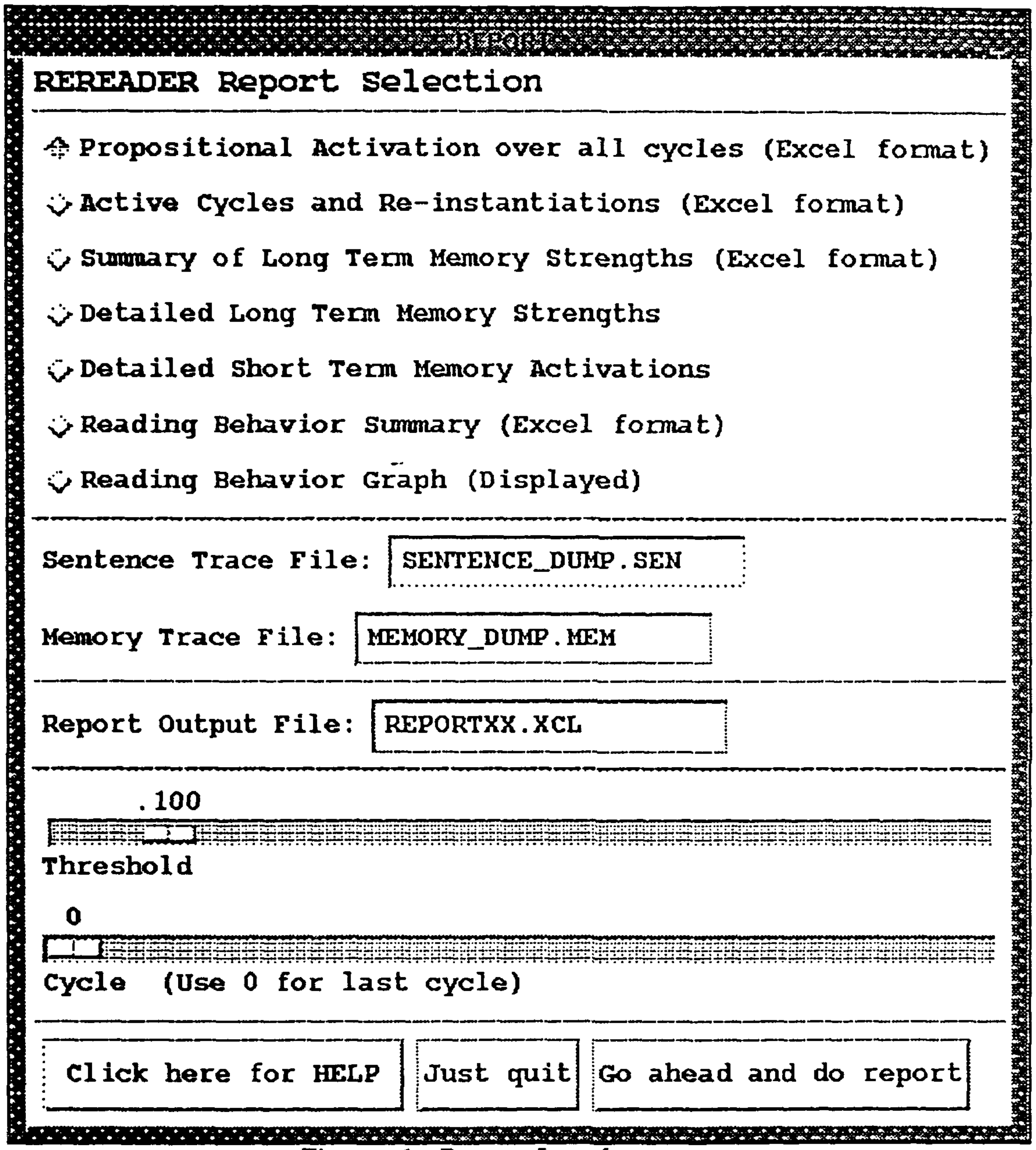
Path File: path.pth

Activation Cap: 50

Motivation: 1

Click here to change Constituent Activations
Click here to change Link Strategy Options
Click here to change Coherence Criteria Options
Click here for HELP
Quit
Go on and Simulate

Figure 5: Main Window for ReReader Simulator


REPORT

REREADER Report Selection

- [✓ Propositional Activation over all cycles (Excel format)]
- Active Cycles and Re-instantiations (Excel format)
- Summary of Long Term Memory Strengths (Excel format)
- Detailed Long Term Memory Strengths
- Detailed Short Term Memory Activations
- Reading Behavior Summary (Excel format)
- Reading Behavior Graph (Displayed)

Sentence Trace File: SENTENCE_DUMP.SEN

Memory Trace File: MEMORY_DUMP.MEM

Report Output File: REPORTXX.XCL

.100
Threshold

0
Cycle (Use 0 for last cycle)

Figure 6 : Report Interface menu